Pengenalan Java

IF2210 – Semester II 2020/2021

by: Yohanes Nugroho; rev: AI, SA, YW, SAR

Mengenal Java

Nama Java

- Nama bahasa pemrograman.
- Nama platform tempat menjalankan program Java, meliputi:
  - Virtual Machine,
  - API (Application Programming Interface).
- Nama Java (konon) diambil dari Kopi Jawa yang sangat terkenal di kalangan pegawai Sun Microsystems.

Sejarah singkat Java

- 1990: James Gosling & timnya di Sun Microsystems memutuskan untuk membuat perangkat lunak khusus untuk *consumer electronic device* (TV, mesin cuci, dll.).
- 1991: Versi awal dikembangkan, dinamai Oak.
- 1993: WWW menjadi populer, web applet dikembangkan. Oak menjadi populer.
- 1995: Nama Oak diganti menjadi Java karena alasan legal.
- 1996: Rilis JDK 1.0.
- 2010: Sun Microsystems dibeli Oracle.
- 2018: Rilis Java SE 10, Java memasuki model *rolling release*.

Five primary goals in the creation of the Java language

- It should be "simple, object oriented, and familiar".
- It should be "robust and secure".
- It should be "architecture neutral and portable".
- It should execute with "high performance".
- It should be "interpreted, threaded, and dynamic".

Bahasa pemrograman Java

- Bahasa Java merupakan bahasa dengan sintaks yang mirip C++ tanpa fitur yang kompleks.
- Umumnya program dalam bahasa Java dikompilasi menjadi bentuk yang dinamakan *bytecode* (tidak dalam bahasa mesin *native*).
  - Seperti bahasa assembly, tapi untuk suatu *virtual machine.*
  - *bytecode* ini dijalankan (*interpret*) di Java Virtual Machine.
- Bahasa Java dirancang sebagai bahasa yang mendukung OOP.

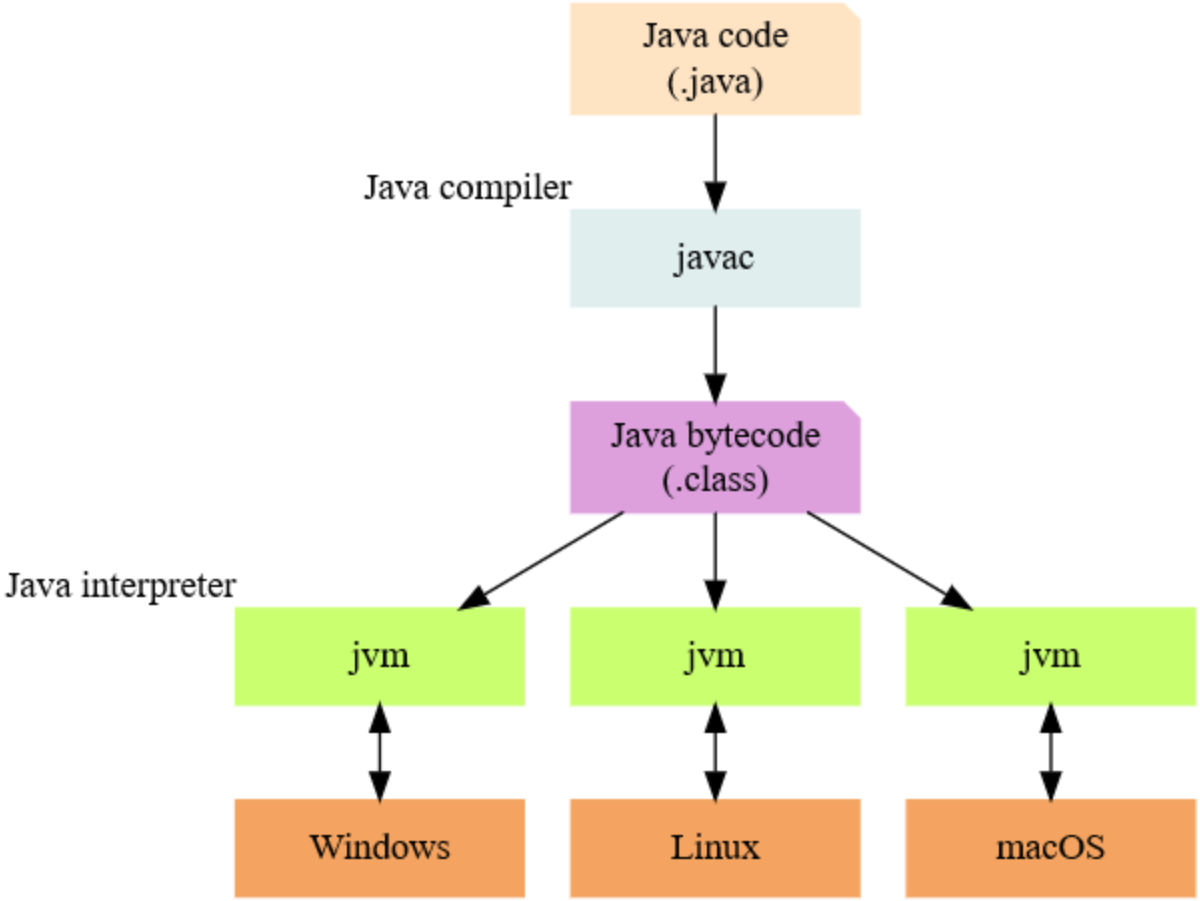
Bahasa Java

- Platform independent
- Object-oriented
- Strongly-typed
- Interpreted and compiled
- Automatic memory management

Java Virtual Machine (JVM)

- JVM adalah suatu program yang menjalankan program Java.
  - Tepatnya, JVM menjalankan *bytecode* dengan menginterpretasi *bytecode*.
- Jika tersedia JVM untuk suatu sistem operasi atau *device* tertentu, maka Java bisa berjalan di sistem komputer tersebut.
- Semboyan Java: *"Write Once Run Anywhere".*

Total platform independence



Java API

- Sekumpulan kelas yang merupakan bagian dari Java Development Kit (JDK).
- Termasuk:
  - Collection
  - GUI programming
  - Networking programming
  - manajemen input & output
  - penanganan database
  - dll.

Lingkup kuliah

- Meliputi:
  - Bahasa Pemrograman Java
    - Pembahasan didasarkan pada konsep OOP yang sudah diberikan.
  - Sedikit API Java
- Sedangkan yang tidak diajarkan:
  - Internal JVM.
  - API Java yang kompleks
  - Pemrograman Java untuk server

Bahan Bacaan

- Spesifikasi Bahasa Java (The Java Language Specification)
- Java Tutorial
- Dokumentasi API Java
- Semua bisa dilihat di java.com

Mengenal lingkungan pemrograman Java

Hello world!

```
// file HelloWorld.java
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello world!");
    }
}
```

- Method statik `main` adalah *entry point* suatu program Java.
- Konvensi: nama *file* harus sama dengan nama kelas, satu *file* berisi satu kelas.

Kompilasi & menjalankan program Java

- Kompilasi:

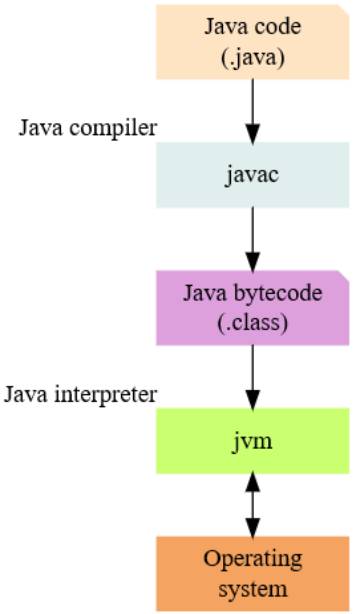
```
javac HelloWorld.java
```

  - Kompilasi menghasilkan `HelloWorld.class`

- Menjalankan:

```
java HelloWorld
```

  - **Tanpa** menuliskan *extension* `.class`.



Penjelasan Hello World

- Semua program Java merupakan kumpulan kelas.
  - Program *Hello world* juga merupakan sebuah kelas.
- Pada C/C++ *entry point* adalah fungsi `main`, di Java *entry point* adalah ***method*** `static main` di sebuah kelas.
  - Karena setiap kelas boleh memiliki `main`, maka pada sebuah aplikasi (yang punya banyak kelas) boleh ada banyak `main` (pengguna memilih `main` yang mana yang dijalankan).

Definisi main

- `main` harus didefinisikan sebagai:
  - Method yang `public` (bisa diakses dari luar kelas).
  - Method yang `static` (bisa dijalankan tanpa instans kelas).
  - Mengembalikan `void` (bukan `int` seperti di C).
  - Memiliki parameter `(String arg[])` yang merupakan parameter dari pengguna.
    - Di C: `int main(int argc, char *argv[]);`
    - Array di Java punya informasi panjang: `arg.length` yang menggantikan kegunaan parameter `argc` di C.
    - Elemen dari `arg` sama seperti char `*argv[]` di C.

Mencetak Hello world

- Baris utama:

```
System.out.println("Hello World");
```

- `System` merupakan nama kelas di Java (kelas standar/default).
- `out` merupakan objek statik dari kelas `PrintWriter`.
  - kelas `PrintWriter` akan dijelaskan kemudian pada konsep I/O Java).
- `out` memiliki method `println()` yang mengambil parameter `String`.

Dasar bahasa Java

Tipe dasar/primitif

- Tipe dasar/primitif adalah tipe bawaan bahasa Java yang bukan merupakan sebuah kelas.
- Java memiliki beberapa tipe dasar seperti di C/C++:
  - `int` (32 bit)
  - `long` (64 bit)
  - `byte` (signed 8 bit)
  - `char` (16 bit Unicode, tidak seperti C/C++ yang merupakan 8 bit ASCII)
  - `float`, `double`

Primitif vs. reference

- Primitif:
  - Tipe dasar seperti `int`, `char`, `float`, dst.
  - Memori dialokasikan saat deklarasi variabel.
  - Operasi *assignment* mengakibatkan penyalinan *value*.
- Reference:
  - Mempunyai semantik serupa dengan pointer di C/C++.
  - Memori dialokasikan saat pemanggilan konstruktor dengan operator `new`.
  - Operasi *assignment* hanya menyalin *reference*, tetap mengacu pada objek yang sama.

```java
class Value { int val; }

class Test {
    public static void main(String args[]) {

        //primitif
        int i1 = 3;
        int i2 = i1; // i1 & i2 variabel berbeda dengan nilai sama
        i2 = 4;
        System.out.print("i1 == " + i1);
        System.out.println(" but i2 == " + i2);

        // reference
        Value v1 = new Value();
        v1.val = 5;
        Value v2 = v1; // v1 & v2 mengacu pada objek yang sama
        v2.val = 6;
        System.out.print("v1 == " + v1.val);
        System.out.println(" and v2 == " + v2.val);
    }
}
```

Output:
```
i1 == 3 but i2 == 4
```

Output:
```
v1 == 6 and v2 == 6
```

Rentang nilai tipe primitif

Tipe	Rentang nilai
`byte`	`-128 .. 127` (8 bits)
`short`	`-32768 .. 32767` (16 bits)
`int`	`-2147483648 .. 2147483647` (32 bits)
`long`	`-9223372036854775808 .. 9223372036854775807` (64 bits)
`float`	`+/-10`$^{-38}$ to `+/-10`$^{+38}$ and `0`, about 6 digits precision
`double`	`+/-10`$^{-308}$ to `+/-10`$^{+308}$ and `0`, about 15 digits precision
`char`	Unicode characters (generally 16 bits per character)
`boolean`	true or false

Pendefinisian tipe & variabel

- Variabel harus dideklarasikan & dialokasikan sebelum digunakan.
- Deklarasi: menyatakan tipe variabel.
  - Contoh: `int x; String str;`
- Alokasi: menyediakan area memori untuk menampung isi variabel.
  - Variabel bertipe non-primitif harus alokasi eksplisit dengan pemanggilan konstruktor.
    - Kecuali `String` – dapat dilakukan

      `String str = "Hello";`

      tidak harus `str = new String("Hello");`

Operator dalam Bahasa Java

- Sifat operator Java sebagian besar sama dengan C/C++:
  - Lihat slide berikut.
- Operator berikut ini hanya ada di Java (tambahan):
  - Perbandingan: `instanceof`
  - Bit: `>>>` (*unsigned shift*)
  - Asignment: `>>>=`
- String: `+` penggabungan string.
- Operator baru yang ada di Java hanya sedikit dan jarang dipakai (kecuali penggabungan string dengan +), sehingga tidak perlu khawatir akan lupa.

Operator Java yang sama dengan C/C++

- Matematik: `+ - * / %` (modulus), unary `+ -`
- Perbandingan: `== != < > <= >=`
- Boolean: `&& || !`
- Bit: `& | ~ << >>`
- Ternary: `cond? true_expr : false_expr`
- Asignment: `= += -= *= /= <<= >>= &= |=`

Operator baru (dibanding C++)

- Ada dua operator yang baru (jika dibandingkan dengan C++) yaitu `>>>` dan `instanceof`.

  `>>>`

  - *unsigned shift right*, *sign* bilangan (bit terkiri) juga di-*shift* ke kanan.

  `x instanceof b`

  - `true` hanya jika `x` (objek) adalah *instance* dari kelas `b`.

Operasi perbandingan pada primitif

- Operator perbandingan (==, <, >, dll) nilai primitif membandingkan nilai primitif tersebut.
- Sifatnya sama dengan C/C++.
- Contoh:

```
int a = 5;
int b = 5;
if (a==b) { /*Sama*/ }
```

Operasi perbandingan pada objek

- Operator `==` terhadap reference membandingkan reference (bukan isi objek).
- Method `equals()` digunakan untuk membandingkan kesamaan isi objek (termasuk juga objek `String`).

```
String a = "Hello";
String b = "World";
if (a.equals(b)) { /*String sama*/}
```

- **Jangan** membandingkan `String` dengan operator `==`.

Kondisional

- Java memiliki sintaks `if` dan `switch` yang sama dengan C/C++
- Di Java `int` tidak sama dengan `boolean`, sehingga hal berikut tidak boleh dilakukan:

```
int a = 1;
if (a) return;
//integer tdk bisa dikonversi ke boolean
```

- Di Java seharusnya:

```
if (a!=0) return;
```

Perulangan

- Java memiliki sintaks *loop* `while`, `for`, `do..while` yang sama dengan C/C++:
  - Perlu diingat bahwa `boolean` tidak sama dengan `int` di Java
- *Enhanced for-loop* (mulai Java 5) untuk *collection* dan *array*:

```
int [] numbers = {2, 3, 5, 7, 11};
int sum;
for(int x: numbers) {
    sum += x;
}
```

Semua di Java adalah *reference*

- Java tidak mengenal *pointer*
  - Semua operasi dan operator pointer yang ada di C/C++ tidak bisa dilakukan di Java (&, *, aritmatika *pointer*)
- Semua objek di Java berlaku sebagai *reference* (sifatnya mirip *pointer*, tapi tanpa * dan &)
- Objek tidak bisa dipertukarkan dengan tipe dasar.
  - Tapi mulai Java 5 ada *auto-boxing/-unboxing* dengan *wrapper class*.

Tipe primitif dan wrapper class

```
boolean → Boolean
   byte → Byte
   char → Character
  float → Float
    int → Integer
   long → Long
  short → Short
 double → Double
```

```java
// Create wrapper object for each primitive type
    Boolean refBoolean = new Boolean(true);
    Byte refByte = new Byte((byte)123);
    Character refChar = new Character('x');
    Short refShort = new Short((short)123);
    Integer refInt = new Integer(123);
    Long refLong = new Long(123L);
    Float refFloat = new Float(12.3F);
    Double refDouble = new Double(12.3D);

// Retrieving the value in a wrapper object
    boolean bool = refBoolean.booleanValue();
    byte b = refByte.byteValue();
    char c = refChar.charValue();
    short s = refShort.shortValue();
    int i = refInt.intValue();
    long l = refLong.longValue();
    float f = refFloat.floatValue();
    double d = refDouble.doubleValue();
```

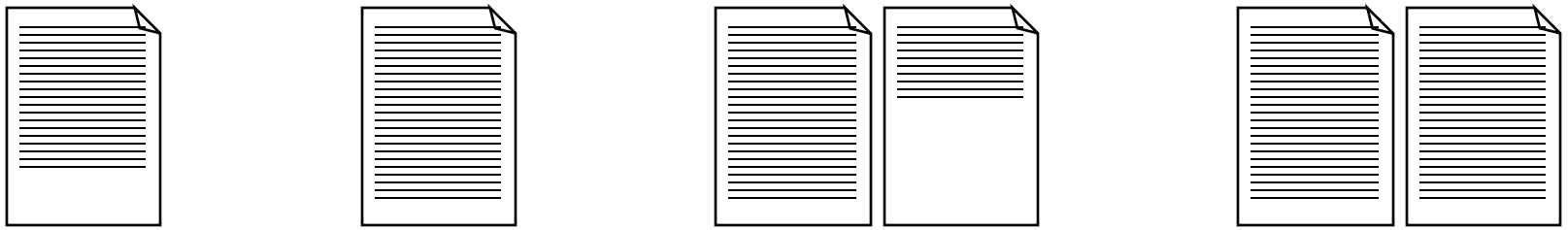
(tidak diperlukan lagi dengan adanya auto-boxing dan -unboxing)

```
// Automatically box primitive
    Boolean refBoolean = true;
    Byte refByte = (byte)123;
    Character refChar = 'x';
    Short refShort = (short)123;
    Integer refInt = 123;
    Long refLong = 123L;
    Float refFloat = 12.3F;
    Double refDouble = 12.3D;

// Automatically unbox wrapper object
    boolean bool = refBoolean;
    byte b = refByte;
    char c = refChar;
    short s = refShort;
    int i = refInt;
    long l = refLong;
    float f = refFloat;
    double d = refDouble;
```

Tugas Baca

- Gast, H. (2015). *How to use objects: code and concepts*. Addison-Wesley Professional.
- Section 1.1 sampai 1.3
- Buat summary min. 1 halaman, max. 2 halaman.
- Pengumpulan: Google Classroom



Not OK OK OK OK